```
# -*- coding: utf-8 -*-
"""
Created on Sat Sep 24 01:10:05 2022

@author: Jun Seok Song (2022 Fall 5)
"""

# Boston house price

 # Data scrapping

import urllib3
import re
address = 'http://lib.stat.cmu.edu/datasets/boston'
http = urllib3.PoolManager()
response = http.request('GET', address)
webContent = str(response.data)


tableContentStart = webContent.find('0.00632')
tableContent = webContent[tableContentStart:]
tableData = re.findall('\d+\.\d+|\d+',tableContent)


tableData
tableData[0:14]
len(tableData)

CRIM = tableData[0::14]
ZN = tableData[1::14]
INDUS = tableData[2::14]
CHAS = tableData[3::14]
NOX = tableData[4::14]
RM = tableData[5::14]
AGE = tableData[6::14]
DIS = tableData[7::14]
RAD = tableData[8::14]
TAX = tableData[9::14]
PTRATIO = tableData[10::14]
B = tableData[11::14]
LSTAT = tableData[12::14]
MEDV = tableData[13::14]


 # Making a dataframe
import pandas as pd
import numpy as np


df = {'CRIM' : pd.Series(CRIM),
'ZN' : pd.Series(ZN),
'INDUS' : pd.Series(INDUS),
'CHAS' : pd.Series(CHAS),
'NOX' : pd.Series(NOX),
'RM' : pd.Series(RM),
'AGE' : pd.Series(AGE),
'DIS' : pd.Series(DIS),
'RAD' : pd.Series(RAD),
'TAX' : pd.Series(TAX),
'PTRATIO' : pd.Series(PTRATIO),
'B' : pd.Series(B),
'LSTAT' : pd.Series(LSTAT),
'MEDV' : pd.Series(MEDV)
}
tableData1 = pd.DataFrame(df)
print(tableData1)



tableData1.head()
tableData1.describe(include='all')
tableData1.shape

for colName in tableData1:
    print(colName)

tableData1.dtypes


 # Changing data types

tableData11 = tableData1.astype(float)
tableData11['RAD'] = tableData11['RAD'].astype(int)
tableData11['CHAS'] = tableData11['CHAS'].astype(int)
tableData11.dtypes


 # Deleting 'B' column

del tableData11['B']

tableData11.head()
tableData11.shape

for colName in tableData11:
    print(colName)

 # Checking out missing values

tableData11.isnull().sum()


 # Describing data

tableData11.describe(include='all')

tableData11['CRIM'].describe(include='all')
tableData11['ZN'].describe(include='all')
tableData11['INDUS'].describe(include='all')
tableData11['NOX'].describe(include='all')
tableData11['RM'].describe(include='all')
tableData11['AGE'].describe(include='all')
tableData11['DIS'].describe(include='all')
tableData11['TAX'].describe(include='all')
tableData11['PTRATIO'].describe(include='all')
tableData11['LSTAT'].describe(include='all')
tableData11['MEDV'].describe(include='all')
```

```
tableData11['CHAS'].unique()
tableData11['RAD'].unique()

 # Box plot

import matplotlib.pyplot as plt
import seaborn as sns

fig, axs = plt.subplots(ncols=7, nrows=2, figsize=(15, 10))
index = 0
axs = axs.flatten()
for i,j in tableData11.items():
    sns.boxplot(y=i, data=tableData11, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)



 # Pairplot (scatter plot + histogram)
sns.pairplot(tableData11)


 # Correlation coefficient
print(tableData11.corr())

 # Heatmap (Correlation coefficient)
plt.figure(figsize = (15,15))
sns.heatmap(tableData11.corr(), vmin=-1, vmax=1, annot=True)



 # Dividing data into training data and test data

np.random.seed(10)
numberRows = len(tableData11)
randomlyShuffledRows = np.random.permutation(numberRows)
randomlyShuffledRows

trainingRows = randomlyShuffledRows[0:405]
testRows = randomlyShuffledRows[405:]


xTrainingData = tableData11.iloc[trainingRows,0:-1]
yTrainingData = tableData11.iloc[trainingRows,-1]

xTestData = tableData11.iloc[testRows,0:-1]
yTestData = tableData11.iloc[testRows,-1]


 ## Linear regression

from sklearn import linear_model

reg = linear_model.LinearRegression()
reg.fit(xTrainingData,yTrainingData)


print(reg.coef_)
 # -0.125, 0.033, 0.001, 2.971(CHAS),
 # -15.271(NOX), 3.942(RM), -0.004, -1.360,
 #  0.306, -0.014, -0.956, -0.543

print(reg.intercept_)
 # 38.753


 ## MSE (Mean squared error)

yPredictions = reg.predict(xTestData)
errors = (yPredictions-yTestData)

from sklearn.metrics import mean_squared_error
mse = mean_squared_error(yTestData,yPredictions)

print(mse)
 # 25.763

 ## r_squared

from sklearn.metrics import r2_score
r2 = r2_score(yTestData,yPredictions)

print(r2)
 # 0.6581

 ## adjusted r_squared

adj_r2 = 1 - (1-r2)*(len(xTestData)-1)/(len(xTestData)-len(xTestData.columns)-1)
print(adj_r2)
 # adj_r2 = 1 - (1-r2)(N-1)/(N-P-1)
          ## N = sample size / P = number of independent variables
 # 0.6115
```